An Approach to RTL False Path Mapping Using Uniqueness of Paths

Hiroshi Iwata Satoshi Ohtake Hideo Fujiwara

Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
Tel: +81-743-72-5226
Fax: +81-743-72-5229
e-mail: {hiroshi-i, ohtake, fujiwara}@is.naist.jp

Abstract— Information on false paths in a circuit is useful for design and test. Since identification of the false paths at gate level is hard, several methods using high-level design information have been proposed. These methods are effective only if the correspondence between paths at register transfer level (RTL) and at gate level can be established. Until now, the correspondence has been established only by some restricted logic synthesis. In this paper, we propose a method for mapping RTL false paths to their corresponding gate level paths without such a specific logic synthesis.

I. INTRODUCTION

For design and test of circuits, false path information is very valuable since it can be used for reducing the time required for logic synthesis, test generation and test application, and circuit area while also minimizing over-testing. From the perspective of design, since design constraints on false paths can be ignored, designers can replace gates on the false paths by smaller gates with larger delay. Furthermore, optimizing paths longer than the critical path can be skipped if they are identified as false paths since they don't have to meet design constraints. Therefore, circuit area and time required for logic synthesis can be made small by using false path information. From the testing point of view, since no test pattern can be generated for path delay faults on false paths, prior false path identification can greatly reduce ATPG time. Furthermore, since some path delay faults on false paths can become testable due to application of design for testability (DFT) and result in over-testing, this can be alleviated by false path identification.

Several false path identification methods at gate level for combinational circuits[1, 2, 7] and for sequential circuits[3, 9] have been proposed. However, since it is difficult to apply false path identification methods at gate level for large circuits containing a tremendous number of paths, some methods using register transfer level (RTL) design information, instead of gate level, have been proposed[5, 10, 11]. While not specifically targeting false paths, Nourani et al.[5] proposed a method using timing analysis and RTL design information to determine the actual critical path and avoid false paths longer than the true critical path. Yoshikawa et al.[10, 11] defined RTL false path and proposed a method to identify them. However, these methods would be useful only if the correspondence between paths at RTL and paths at gate level can be established. Until now, the correspondence has been established through module interface preserving-logic synthesis (MIP-LS)[10]. Currently, using MIP-LS is the only way to guarantee information of the correspondence. However, it is not practical to restrict synthesis only to MIP-LS.

In this paper, we focus on path mapping from a set of RTL false paths to gate level paths without considering MIP-LS. We first propose a method of mapping a set of RTL paths to its corresponding gate level paths (this is called path mapping) with an arbitrary logic synthesis independent of false paths. The proposed method maps RTL signal lines composing the RTL paths to gate level nets by using functional equivalence relation of signal lines (this is called signal line mapping). The effort required for signal line mapping is alleviated by using the uniqueness of a set of the RTL paths and the rough candidate selection method. Since the number of signal lines that uniquely identify a set of RTL paths is much lower than that of whole signal lines in the set of RTL paths and our path mapping algorithm only needs to map the reduced signal lines, the number of RTL signal lines to be mapped is significantly reduced. Signal line mapping is achieved by checking the equivalence between signal lines and all the gate level nets: however, it is obviously not practical. Therefore, we use the method which finds candidates of the functionally equivalent nets from a gate level circuit by using diagnostics technique[6].

Since the gate level paths mapped by our method are represented as signal lines, each gate level path does not need to be fully specified as a path, so we are able to handle bounded paths. This representation is compatible with EDA tools, like Synopsys Design Constraint (SDC). Experimental results show that many RTL paths can be mapped to gate level paths using the proposed method within a reasonable time.

Then, we consider false path mapping. The definition of RTL false path in [10] assumes MIP-LS and the assumption guarantees that the corresponding gate level paths are false. In this paper, we show that any corresponding gate level path that is mapped from the set of RTL false paths by using the proposed method with an arbitrary logic synthesis is false. Experimental results show that our path mapping method can establish the correspondences of RTL false paths and many gate level false paths.

The rest of this paper is organized as follows. Preliminaries are presented in Section II. Section III presents the proposed RTL path mapping method. Section IV shows that the gate level paths mapped from a set of RTL false paths with the proposed method are false. Experimental results are given in Section V. Section VI concludes the paper.

II. Preliminaries

A. Circuit model

In this paper, we only consider structural RTL designs. A structural RTL design consists of a controller represented by a combinational module and a state register, and a datapath represented by RTL modules and signal lines connecting them where an RTL module is an operational module, a register or a MUX and a signal line has an arbitrary bit width.

B. Gate level and RTL path representation

Definition 1: *(Gate level path)* An ordered set of gate level nets $\{e_1^G, \ldots, e_n^G\}$ is called a *gate level path* if it satisfies the following conditions.

1. e_1^G is the net directly connected to a primary input or the output of an FF
2. e_n^G is the net directly connected to a primary output or the input of an FF
3. $e_i^G(i = 2, \ldots, n-1)$ is the net connecting between the gates having e_{i-1}^G as an input and e_{i+1}^G as an output, respectively □

Definition 2: *(Sub gate level path)* A subset of a gate level path p^G is called a *sub gate level path of* p^G. □

Definition 3: *(RTL path)* An ordered set of RTL signal lines $\{e_1^R, \ldots, e_n^R\}$ is called an *RTL path* if it satisfies the following conditions.

1. e_1^R is the RTL signal line directly connected to a primary input or the output of a register
2. e_n^R is the RTL signal line directly connected to a primary output or the input of a register
3. $e_i^R(i = 2, \ldots, n-1)$ is the RTL signal line connecting between the modules having e_{i-1}^R as an input and e_{i+1}^R as an output, respectively □

Definition 4: *(Sub RTL path)* A subset of an RTL path p^R is called a *sub RTL path of* p^R. □

An RTL signal line consists of one-bit signal lines as follows.

Definition 5: *(Bit-sliced RTL signal line)* For an RTL signal line s, each one bit signal line separated from s is referred to as a *bit-sliced RTL signal line of* s. The i-th bit of s is represented as $s[i]$. □

Definition 6: *(Bit-sliced RTL path)* An ordered set of bit-sliced RTL signal lines $\{e_1^R[k_1], \ldots, e_n^R[k_n]\}$ is called a *bit-sliced RTL path* if it satisfies the following conditions.

1. $e_1^R[k_1]$ is the k_1-th bit-sliced RTL signal line directly connected to a primary input or the output of a register
2. $e_n^R[k_n]$ is the k_n-th bit-sliced RTL signal line directly connected to a primary output or the input of a register
3. $e_i^R[k_i](i = 2, \ldots, n-1)$ is the k_i-th bit-sliced RTL signal line connecting between the modules having $e_{i-1}^R[k_{i-1}]$ as an input and $e_{i+1}^R[k_{i+1}]$ as an output, respectively □

Definition 7: *(Sub bit-sliced RTL path)* A subset of a bit-sliced RTL path p^R is called a *sub bit-sliced RTL path of* p^R. □

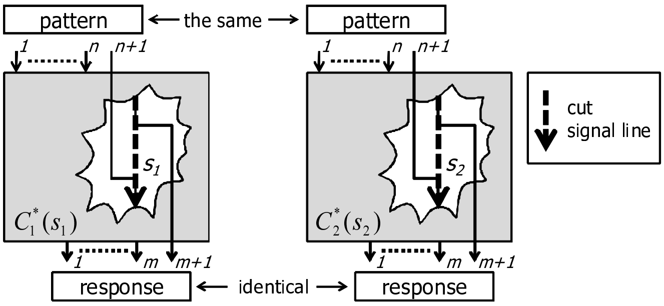

Fig. 1. Functionally equivalent signal lines s_1 and s_2.

C. Relation between signal lines

Here, we first define *signal line cutting*, which is an operation needed for defining functionally equivalent signal lines.

Definition 8: *(Signal line cutting)* For a combinational circuit C with n inputs, m outputs and an internal signal line s, the following operation is referred to as *cutting* C *on* s.

1. Create the $(n+1)$-th new input port and the $(m+1)$-th new output port.
2. Remove the signal line s.
3. Create connections between $(n+1)$-th input port and the end point of s and between the start point of s and the $(m+1)$-th output port. □

In the following discussion, we represent the combinational circuit resulting from the above operations as $C^*(s)$.

For two functionally equivalent combinational circuits, we define a functional equivalence of signal lines as follows.

Definition 9: *(Functionally equivalent signal line)* For two functionally equivalent combinational circuits C_1 and C_2 with internal signal lines s_1 and s_2, respectively, s_1 *and* s_2 *are functionally equivalent* if and only if $C_1^*(s_1)$ and $C_2^*(s_2)$ are functionally equivalent. □

In the following discussion, we represent the relation of functional equivalence between signal lines s_1 and s_2 as $s_1 \equiv_l s_2$.

Figure 1 illustrates functionally equivalent signal lines. The signal lines s_1 and s_2 are functionally equivalent if the responses from $C_1^*(s_1)$ and $C_2^*(s_2)$ are identical for any input pattern.

D. Relation between paths

We define the functional equivalence between the sub bit-sliced RTL path and sub gate level path as follows.

Definition 10: *(Functionally equivalent path)* Sub bit-sliced RTL paths and sub gate level paths are simply referred to as *sub paths*. Sub paths $q_1 = \{e_{1_1}, \ldots, e_{1_n}\}$ and $q_2 = \{e_{2_1}, \ldots, e_{2_m}\}$ are *functionally equivalent* if q_1 and q_2 satisfy the following conditions.

1. $n = m$
2. $e_{1_i} \equiv_l e_{2_i}$ $(i = 1, \ldots, n)$ $\square$

For mapping a given RTL path to gate level paths, it is sufficient to map only the RTL signal lines that uniquely identify the RTL path to gate level nets. Therefore we provide the following definition to alleviate the signal line mapping effort.

Definition 11: *(Identification of path)* A sub RTL path q^R is said to *uniquely identify* an RTL path p^R if p^R is the only path that properly includes q^R. $\square$

Definition 12: *(Identification of path set)* A sub RTL path q^R is said to *uniquely identify* a set of RTL paths, P^R, if P^R is the only set of RTL paths that properly includes q^R. $\square$

III. Proposed method of path mapping

In this section, we formulate the path mapping problem and present a solution of the problem independent of false path. Consideration of false paths is described in Section IV.

A. Path mapping problem

We formulate the path mapping problem as a problem to find a set of gate level paths corresponding to a set of RTL paths.

For solving path mapping problem, it is sufficient to consider only the RTL combinational circuit C^R which is the combinational part of a given structural RTL design S^R and the gate level combinational circuit C^G which is the combinational part of a gate level design synthesized from S^R. We assume that there exists a one-to-one correspondence between the input/output signal lines of C^R and C^G. This relation is called *I/O mapping information*. The I/O mapping information of C^R and C^G can be obtained by preserving all the bits of the registers in S^R during logic synthesis. The preservation is common for logic synthesis for structural RTL designs.

Definition 13: *(Path mapping problem)*

Input
- C^R: an RTL combinational circuit
- C^G: a gate level circuit that is functionally equivalent to C^R
- The I/O mapping information between C^R and C^G
- P^R: a set of RTL paths

Output $P^G = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m_i} P_{ij}^G$

Where P_{ij}^G is defined as follows. Let $q_i^R (i = 1, \ldots, n)$ be a sub RTL path that uniquely identifies P^R and $q_{ij}^R (j = 1, \ldots, m_i)$ be a sub bit-sliced RTL path of q_i^R where n is the number of the sub RTL paths of p^R. Let q_{ij}^G be a sub gate level path that is functionally equivalent to q_{ij}^R. P_{ij}^G is a set of gate level paths including q_{ij}^G. $\square$

B. Path mapping algorithm

We propose an algorithm solving the path mapping problem as follows. The algorithm establishes correspondences between a set of RTL paths, P^R, and a set of gate level paths, P^G.

1. Generate the minimum sub RTL path $q_i^R (i = 1, \ldots, n)$ that uniquely identifies P^R.
2. Try to obtain a gate level net $e_{ij_k}^G$ that is functionally equivalent to each bit-sliced RTL signal line $e_{ij_k}^R (k = 1, \ldots, l)$, where $e_{ij_k}^R$ is an element of a bit-sliced RTL path $q_{ij}^R (j = 1, \ldots, m_i)$ of q_i^R, m_i is the number of combinations of bit-sliced RTL paths obtained by specifying bit portion of every RTL signal line on q_i^R, and l is the number of RTL signal lines on q_i^R. Go to step 3 if $e_{ij_k}^R$ or $i \geq n$.
3. For each pair of i and j, find all the gate level paths that include $\{e_{ij_1}^G, \ldots, e_{ij_l}^G\}$. The set of the obtained gate level paths is referred to as P_{ij}^G.
4. Calculate $P^G = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m_i} P_{ij}^G$.

Notice that, the signal line mapping in step 2 is described in the next subsection. We assume that at most one gate level net is functionally equivalent to a bit-sliced RTL signal line for simplifying the algorithm description. In our experiments reported in Section V, we did not face a case that more than one gate level net is mapped. However, we can handle multiple nets by taking all the paths that through the nets into account. In step 3 and 4, all the gate level paths are not needed to be listed. It is not practical to list gate level paths. Instead of listing, paths are represented by just specifying nets, $\{e_{ij_1}^G, \ldots, e_{ij_l}^G\}$, which are passed through. This representation is compatible or familiar with EDA tools, like SDC description.

C. Signal line mapping

In this section, we formulate the problem finding functionally equivalent nets. Then, we will show an algorithm for solving the problem. Signal line mapping algorithm is used in the proposed path mapping algorithm.

C.1. Signal line mapping problem

We formulate the *signal line mapping problem* to find a set of nets, which is functionally equivalent to a bit-sliced RTL signal line in an RTL circuit, in a gate level circuit.

Definition 14: *(Signal line mapping problem)*

Input
- C^R: an RTL combinational circuit
- C^G: a gate level circuit that is functionally equivalent to C^R
- The I/O mapping information between C^R and C^G
- $e^R[k]$: the k-th bit-sliced RTL signal line of an RTL signal line e^R in C^R

Output $E^G = \{e^G | e^G \equiv_l e^R[k]\}$ where e^G is a net in C^G $\square$

C.2. Signal line mapping algorithm

Given an RTL combinational circuit C^R and a gate level combinational circuit C^G, checking functional equivalence between a bit-sliced RTL signal line $e^R[k]$ in C^R and a gate level net e^G in C^G can be performed by applying all the possible input patterns to both circuits $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$, and comparing their output responses. It is achieved by applying equivalence checking[4, 8] for $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$.

However, it is not practical to explicitly check the functional equivalence for all the possible combinations between $e^R[k]$ and e^G in C^G.

Ravi et al.[6] proposed a method finding candidates for functionally equivalent nets of a given bit-sliced RTL signal line using fault diagnosis techniques. In this paper, their method is utilized to solve the signal line mapping problem. More specifically, their method injects a stuck-at fault on the bit-sliced RTL signal line and finds the stuck-at faults, which have the identical behavior of the fault under the test patterns, in the gate level circuit. The faults in the gate level circuit and the fault in the RTL circuit are said to be *equivalent*. A necessary condition of functional equivalence is that the responses of the RTL circuit and the gate level circuit are identical when value v's are fixed to $e^R[k]$ and e^G, respectively (see Fig.2 (b)). It is the same situation when s-a-v faults are assumed to be presented on $e^R[k]$ and on e^G, respectively (see Fig.2 (a)). To make our signal line mapping algorithm complete, we perform functional equivalence checking for $e^R[k]$ and each above mentioned candidate net e^G. The overall algorithm to solve the signal line mapping problem is shown in the following.

1. Generate a complete test set T for all the testable stuck-at faults in C^G.
2. For each $v \in \{0,1\}$, the following two steps are performed.
   (a) Obtain a set of faulty output responses R_{fv} by applying T to the RTL circuit C^R with an injected s-a-v fault on the given bit-sliced RTL signal line $e^R[k]$.
   (b) Find all the single s-a-v faults of C^G such that all the faulty circuits induced by the faults respond the same output responses R_{fv} when T is applied to these circuits. A set of the nets having equivalent faults is referred to as E^{Gv}.
3. Obtain $E^G = E^{G0} \cap E^{G1}$.
4. For each $e^G \in E^G$, create $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$ by cutting C^R and C^G on $e^R[k]$ and e^G, respectively.
5. Perform equivalence checking for $C^R(e^{R*}[k])$ and $C^{G*}(e^G)$ and eliminate e^G from E^G if they are not functionally equivalent.

Steps 1 to 3 are the same as the procedure for finding functionally equivalent signal line by using fault diagnosis technique in [6]. In [6], the complete test set T for the detectable faults in a gate level circuit is used as the input patterns for fault diagnosis. The procedure first finds s-a-0 (resp. 1) faults in C^G that are equivalent to the s-a-0 (resp. 1) fault injected on $e^R[k]$ under the test set T. Then the procedure selects gate level nets that have both s-a-0 and s-a-1 faults as the candidates of equivalent nets. These nets obtained by the steps satisfies the necessary condition of the functional equivalence. Finally, steps 4 and 5 are performed to guarantee sufficiency.

The completeness of the overall algorithm is shown in Theorem 1. Here we assume that the fault diagnosis technique used in the algorithm can report all the suspected faults, i.e., it never miss equivalent fault under its input patterns.

Theorem 1 *Given an RTL combinational circuit C^R, its synthesized gate level circuit C^G and a bit-sliced RTL signal line $e^R[k]$ in C^R. Any $e^G \in E^G$ is functionally equivalent to $e^R[k]$ if and only if E^G is the set of gate level nets obtained by the signal line mapping algorithm.* □

[Proof] First, we show that Steps 1 to 3 guarantee that the primary outputs of C^G and C^R have the same response for any input pattern in T when e^G and $e^R[k]$ have the same value which is the necessary condition of functional equivalence. Let C^R has n inputs ($x^R[i](i = 1, \ldots, n)$) and m outputs ($z^R[i](i = 1, \ldots, m)$). $C^{R*}(e^R[k])$ has $n+1$ inputs ($x^{R*}[i](i = 1, \ldots, n+1)$) and $m+1$ outputs ($z^{R*}[i](i = 1, \ldots, m+1)$). We inject a s-a-$v$ fault on $e^R[k]$ in C^R where $v \in \{0,1\}$. For any $t \in T$, the output response from $z^R[1], \ldots, z^R[m]$ of C^R with the s-a-v obtained by applying t to C^R with the fault and that from $z^{R*}[1], \ldots, z^{R*}[m]$ of $C^{R*}(e^R[k])$ obtained by applying $t\&v$ to C^{R*} are identical where "$a\&b$" denotes concatenation of vectors a and b.

Let C^G has n inputs ($x^G[i](i = 1, \ldots, n)$) and m outputs ($z^G[i](i = 1, \ldots, m)$). $C^{G*}(e^G)$ has $n+1$ inputs ($x^{G*}[i](i = 1, \ldots, n+1)$) and $m+1$ outputs ($z^{G*}[i](i = 1, \ldots, m+1)$). We inject a s-a-$v$ fault on signal line e^G in C^G. For any $t \in T$, the output response from $z^G[1], \ldots, z^G[m]$ of C^G with the s-a-v obtained by applying t to C^G with the fault and that from $z^{G*}[1], \ldots, z^{G*}[m]$ of $C^{G*}(e^G[k])$ obtained by applying $t\&v$ to C^{G*} are identical.

Consequently, for any input pattern of T, the output responses from $z^{R*}[1], \ldots, z^{R*}[m]$ and $z^{G*}[1], \ldots, z^{G*}[m]$ of $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$, respectively, are the same because C^R and C^G are functionally equivalent and s-a-v faults on $e^R[k]$ and e^G are equivalent under T. It is a necessary condition of functional equivalence between $e^R[k]$ and e^G.

From the assumption of fault diagnose, all e^G are able to get as E^G where e^G satisfies the above conditions. It is obvious that E^G properly include all the gate level nets that are functionally equivalent to $e^R[k]$. Therefore we only need to show that the steps 4 and 5 can exclude nets if and only if the nets are not functionally equivalent to $e^R[k]$. Clearly, e^G is eliminated from E^G if e^G is not functionally equivalent to $e^R[k]$. Otherwise, it is not eliminated. Thus the theorem holds true. □

IV. RTL FALSE PATH MAPPING

In [12], Yoshikawa et al. defined non-robust untestable paths for RTL circuits as follows.

Definition 15: *(RTL non-robust untestable path)* An RTL path p in an RTL circuit S^R is RTL non-robust untestable (RTL-NRU) if all the gate-level paths in $\delta(p)$ are non-robust untestable (NRU) for any gate-level circuit S^G synthesized from S^R, where $\delta(p)$ is a set of gate level paths corresponding to p. □

In order to guarantee the correspondence between RTL-NRU and $\delta(p)$, restricted logic synthesis called *module interface preserving logic synthesis (MIP-LS)* is employed.

Under the assumption on logic synthesis, they also provide sufficient condition of RTL-NRU based on control signals of MUXes and registers. In this paper, we refer to an RTL path

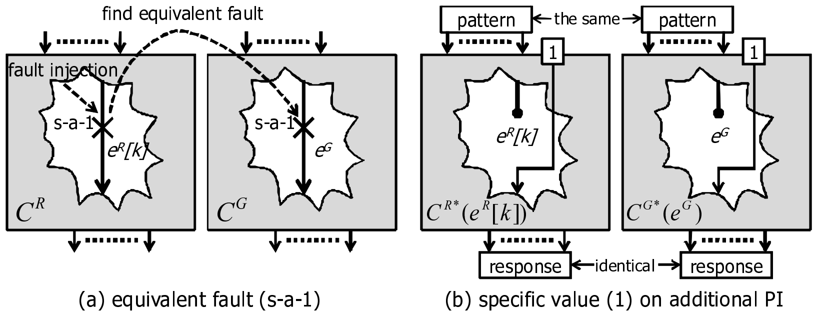

Fig. 2. Relation between equivalent faults and functionality of respective signal lines.

that satisfies the sufficient condition of RTL-NRU as an RTL false path. Non-robust untestable gate level paths for both transitions are also referred to as gate level false paths. For a given path $p = \{e_1, \ldots, e_n\}$ in an RTL circuit, intuitively, the condition is as follows. The path p is RTL false if at least one of the following satisfies for any input sequence and any t: (1) there is no controllability to make a transition on the starting register which drives e_1 in cycles between t and $t+1$ (if any), (2) there is no propagatability of a value from e_i to e_{i+1} for some i $(i = 1 \ldots n)$ in $t+1$ (if any), (3) no response on e_n is captured on the ending register in $t+2$ (if any), and (4) there is no observability for the captured responses (if any). These are checked only by examining control signal values of MUXes and registers supplied from the controller. Notice that, in their RTL circuit model, for an RTL circuit, state transitions of the controller are known and are completely specified for all the pairs of states and input vectors. Detailed description is available in [12].

The condition means that no transition can be propagated through an RTL-NRU path, which is identified based on the condition, in non-robust sensitization criteria or the response captured at the ending register cannot be observed. If we can remove the assumption on logic synthesis, we can utilize the identification method reported in [10] for more general circuits synthesized without the restriction. Therefore, we obtain the following theorem.

Theorem 2 *For an RTL false path p^R, in an RTL circuit S^R, any $p^G \in P^{GA}$ that is mapped from p^R with our path mapping method is gate level false.* □

[Proof] Suppose that p^R and p^G consists of $\{e_1^R, \ldots, e_n^R\}$ and $\{e_1^G, \ldots, e_m^G\}$, respectively, and each RTL signal e_i^R of p^R has arbitrary bits. From sufficient condition of RTL-NRU, for an input sequence, p^R satisfies at least one of the four conditions, as described above. From the assumption of existence of I/O mapping information, functional equivalence of internal signal line and combinational parts of S^R and S^G are functionally equivalent, we can say the following. If it satisfies (1), all the bit-sliced signal lines of e_1^R cannot have transitions in cycles t to $t+1$. Therefore, e_1^G cannot have any transition in t to $t+1$. If it satisfies (3) or (4), values of all the bit-sliced signal lines of e_n^R cannot be observed. Therefore, any values on e_m^G at $t+1$ cannot be observed. If it satisfies (2), all the bit sliced e_n^R cannot have transitions which are from e_1^R and by way of e_i^R (if any) in t to $t+1$, where e_i^R is necessary to be mapped. Therefore, e_m^G cannot have any transition that is from e_1^G and by way of e_j^G (if any) in t to $t+1$, where e_j^G is mapped from e_i^R. Thus, the theorem holds true. □

By this theorem, we are able to treat gate level false paths in a gate level circuit synthesized with an arbitrary logic synthesis (without restricting logic synthesis to MIP-LS) through the proposed path mapping method.

V. Experimental results

In this section, we show experimental results for evaluating our RTL path mapping method by mapping RTL paths and RTL false paths identified with the method proposed in [10]. We used three RTL benchmark circuits, LWF, Tseng and Paulin and an industrial circuit, MPEG. In this experiments, we used only the datapath part of each circuit and tried to map all the paths in the datapath. Table I shows the circuit characteristics of the circuits. Columns "#bit", "#PI", "#PO" and "#reg" show the bit width, the number of primary inputs, that of primary outputs and that of registers, respectively. Sub columns "MIP-LS" and "Arbitrary" under "Area (#gates)" show the circuit area synthesized by MIP-LS[10] and that without restriction, respectively. From the area comparison, we confirmed that our method eliminate the impact on logic synthesis results. In this experiments, we used Synopsys DesignCompiler to perform logic synthesis, Synopsys TetraMax to generate test patterns for gate level circuits synthesized with "Arbitrary", Cadence Encounter Test and Diagnostics as a fault diagnostic engine, Synopsys Formality to perform equivalence checking and Synopsys PrimeTime to enumerate the gate level paths on Sun Microsystems Sun Fire X4100(Opteron 256(3GHz), 16GB memories).

We use the RTL path mapping ratio $Pmr = \frac{|P^{RT}|}{|P^R|} \times 100[\%]$ as an evaluation criterion, where $|P^R|$ is the total number of RTL paths in the datapath and $|P^{RT}|$ is the number of RTL paths mapped. Furthermore, to evaluate more in detail, we consider bit-sliced RTL paths in the datapath. We use the bit-sliced RTL path mapping ratio $Pmr_b = \frac{|P_b^{RT}|}{|P_b^R|} \times 100[\%]$, where $|P_b^R|$ is the total number of bit-sliced RTL paths in

TABLE I
CIRCUIT CHARACTERISTICS.

Circuit	#bit	#PI	#PO	#reg	Area (#gate)	
					MIP-LS	Arbitrary
LWF	16	2	2	5	1,571	1,467
Tseng	8	3	2	6	1,357	1,077
Paulin	8	2	2	7	1,590	1,303
MPEG	8	5	16	241	38,183	28,454

TABLE II
PATH MAPPING RESULTS.

	LWF	Tseng	Paulin	MPEG
Pmr[%]	73.7	90.0	100.0	100.0
Pmr_b[%]	74.2	96.8	100.0	100.0
CPU[sec]	28.14	21.74	0.30	0.10

the datapath and $|P_b^{RT}|$ is the number of bit-sliced RTL paths mapped. Table II shows the path mapping ratios, bit-sliced path mapping ratios and time required for these mapping.

Table III shows the signal line and path mapping results in detail. Rows “#Ptotal”, “#Punique”, “#Stried”, “#Smapped” and “#Pmapped” show the total number of RTL paths, the number of paths uniquely identified with the I/O mapping information, the number of RTL signal lines targeted by signal line mapping, the number of RTL signal lines mapped, i.e., the gate level nets which are functionally equivalent to the bit-sliced RTL signal lines were found, and the number of the RTL paths mapped. Columns “RTL” and “bsRTL” under each circuit name mean that bundled RTL and bit-sliced RTL, respectively. Thanks to Definition 11 (uniquely identification of path), most of the RTL paths were able to be mapped only by using I/O mapping information or CPU time was able to be saved. The proposed method achieved the RTL path mapping ratio and the bit-sliced RTL path mapping ratio of average 90.9% and 92.8%, respectively. The reason why (bit-sliced) RTL paths which were not able to be mapped to gate level paths existed is that the algorithm was not able to find any signal line needed for path mapping, i.e., there exist no functionally equivalent net in the gate level circuits.

Table IV shows the result of RTL false path mapping and time required for these mapping. Rows “#Ptotal”, “#Pfalse”, “Ratio”, “Total”, “Unique”, “Ravi”, “FEchk”, “Pwhole” and “Pfalse” show the total number of paths, the number of false paths, the ratio of #Pfalse to #Ptotal, the total time required for false path mapping, the time required for finding the candidate of functionally equivalent signal line, the time required for equivalence checking, the time required for listing the whole paths in the gate level circuit, and the time required for listing the false paths mapped, respectively. Columns “RTL” and “Gate level” under each circuit name mean that number of paths in RTL and the ones in gate level, respectively. Many gate level false paths was able to be found with our proposed path mapping method in practical time without considering MIP-LS.

Table V shows the signal line and path mapping results in detail. Rows “#Pfalse”, “#Punique”, “#Stried” and “#Smapped” show the number of RTL false paths, the number of paths uniquely identified with the I/O mapping information, the number of bit-sliced RTL signal lines targeted by signal line mapping, and the number of bit-sliced RTL signal lines mapped, respectively.Therefore, we can say that the proposed method finds almost all gate level false paths corresponding to the given RTL false paths.

TABLE V
DETAILS OF THE FALSE PATH MAPPING.

	LWF	Tseng	Paulin	MPEG
#Pfalse	5	6	13	32
#Punique	4	5	13	32
#Stried	32	16	0	0
#Smapped	0	7	-	-

VI. CONCLUSIONS

Establishing correspondence between an RTL circuit and its synthesized gate level circuit is important for high level testing approaches. In this work, we focus on correspondence between paths in the RTL circuit and paths in the gate level circuit. Existence of the correspondence enables the technologies that handle a path at RTL as a bundle of a tremendous number of paths in the gate level circuit. There are methods to identify false paths using RTL design information [10, 11], which is feasible only if RTL paths can be mapped into gate level paths.

In this paper, we proposed a method to establish correspondence between a set of RTL paths and gate level paths without restricting logic synthesis. To the best of our knowledge, this is the first work that tackles RTL to gate level path mapping. Furthermore, we showed that RTL false paths identified by [10] can be mapped to gate level false paths with our proposed method. In our experiments, the proposed path mapping method was utilized as a false path mapping procedure, and many false paths were able to be found in a circuit synthesized with an arbitrary logic synthesis by using our proposed path mapping method.

REFERENCES

[1] K. T. Cheng and H. C. Chen, “Classification and identification of nonrobust untestable path delay faults,” *IEEE Trans. on CAD*, 15(8):845–853, Aug. 1996.

[2] S. Kajihara, K. Kinoshita, I. Pomeranz, and S. Reddy, “A method for identifying robust dependent and functionally unsensitizable paths,” In *International Conference on VLSI Design*, pp. 82–87, 1997.

[3] A. Krstić, S. T. Chakradhar, and K.-T. T. Cheng, “Testable path delay fault cover for sequential circuits,” In *European Design and Test Conference*, pp. 220–226, 1996.

[4] A. Kuehlmann and F. Krohm, “Equivalence checking using cuts and heaps,” In *Proceedings of the 34th annual Design Automation Conference(DAC’97)*, pp. 263–268, 1997.

[5] M. Nourani and C. A. Papachristou, “False path exclusion in delay analysis of RTL structures,” *IEEE Trans. on Very Large Scale Integration Systems*, 10(1):30–43, Feb. 2002.

[6] S. Ravi, I. Ghosh, V. Boppana, and N. K. Jha, “Fault-diagnosis-based technique for establishing RTL and gate-level correspondences,” *IEEE Trans. on CAD*, 20(12):1414–1425, 2001.

TABLE III

SIGNAL LINE AND PATH MAPPING RESULTS IN DETAIL.

	LWF		Tseng		Paulin		MPEG	
	RTL	bsRTL	RTL	bsRTL	RTL	bsRTL	RTL	bsRTL
#Ptotal	19	4,600,384	20	36,448	29	123,600	606	326,176
#Punique	14	3,412,544	18	31,840	29	123,600	606	326,176
#Stried	5	80	5	40	0	0	0	0
#Smapped	0	13	0	12	-	-	-	-
#Pmapped	14	3,415,360	18	35,296	29	123,600	606	326,176

TABLE IV

FALSE PATH MAPPING RESULTS.

	LWF		Tseng		Paulin		MPEG	
	RTL	Gate level	RTL	Gate level	RTL	Gate level	RTL	Gate level
#Ptotal	19	1,845,916	20	856,116	29	2,307,064	606	1,784,824
#Pfalse	5	470,300	6	418,752	13	1,610,968	32	16
Ratio[%]	26.32	25.48	30.00	48.91	44.83	69.83	5.28	0.00
Total[s]	15.36		21.73		0.27		1.72	
Unique[s]	0.21		0.24		0.27		1.72	
Ravi[s]	15.15		17.07		0.00		0.00	
FEchk[s]	0.00		4.42		0.00		0.00	
Pwhole[s]	93.21		37.21		103.39		303.65	
Pfalse[s]	24.26		19.07		73.53		0.22	

[7] S. M. Reddy, S. Kajihara, and I. Pomeranz, "An efficient method to identify untestable path delay faults," In *Asian Test Symposium*, pp. 233–238, 2001.

[8] Synopsys, inc. *Formality User Guide*, c-2009.06 edition, June 2009.

[9] R. Tekumalla and P. Menon, "Identifying redundant path delay faults in sequential circuits," In *International Conference VLSI Design*, pp. 406–411, 1996.

[10] Y. Yoshikawa, S. Ohtake, and H. Fujiwara, "False path identification using RTL information and its application to over-testing reduction for delay faults," In *Asian Test Symposium*, pp. 65–68, 2007.

[11] Y. Yoshikawa, S. Ohtake, T. Inoue, and H. Fujiwara, "Fast false path identification based on functional unsensitizability using RTL information," In *14th Asia and South Pacific Design Automation Conference 2009 (ASP-DAC'09)*, pp. 660–665, Jan. 2009.

[12] Y. Yoshikawa, S. Ohtake, T. Inoue, and H. Fujiwara, "A synthesis method to alleviate over-testing of delay faults based on RTL don't care path identification," In *IEEE VLSI Test Symposium (VTS'09)*, pp. 71–76, May. 2009.